

# Natural Language Parsing

Mattox Beckman

beckman@iit.edu

Illinois Institute of Technology

Objectives for today:

- Have fun.
- Make prolog look like it's smart.
- Know how to write some input routines.

- Prolog has a very simple input model.
- `get0(C)` puts the next character of input into `C`.

```
1 ?- get0(X) .  
2 | : A  
3  
4 X = 65  
5 Yes
```

- To get a whole word, we need to explain what a word is.
- Note the use of pattern matching: `getwd([C|W])`

```
1 eow(10).    10 is carriage-return
2 eow(32).    32 is space
3
4 getwd([C|W]) :- get0(C), not(eow(C)), !, getwd(W).
5 getwd([]).
6
7 ?- getwd(X).
8 |: hello
9
10 X = [104, 101, 108, 108, 111]
```

- A sentence is done when it ends with a period or question mark.

```
1 punct(46).    46 is period
2 punct(63).    63 is question-mark
3
4 lastcharpunct(X) :- reverse(X,[P|_]), punct(P).
5 choplastchar(X,Y) :- reverse(X,[_|Z]), reverse(Z,Y)
6
7 ?- getwd(X), choplastchar(X,Y).
8 |: hi.
9
10 X = [104, 105, 46]
11 Y = [104, 105]
```

```
1 getsentence([W|S]) :-  
2     getwd(WC),  
3     (    (lastcharpunct(WC),  
4         choplastchar(WC,W), !, S=[]);  
5         W=WC, !, getsentence(S)).  
6  
7 ?- getsentence(X).  
8 |: hi there.  
9  
10 X = [[104, 105], [116, 104, 101, 114, 101]]
```

```
1 getslist(X) :- getsentence(S), maplist(name,X,S).  
2  
3 ?- getslist(X).  
4 |: hi there.  
5  
6 X = [hi, there]
```

● Now we have our input predicate.

- We want a mechanism to record knowledge we gain.

```
1 check(L) :- known(L), !,  
2     writef("I already know about that."), nl.  
3 check(L) :- assert(known(L)), assert(L).  
4  
5 forget(L) :- retract(known(L)), retract(L).
```



```
1 go :- dynamic(acldause/1), dynamic(known/1),  
2     writef("Hello.  I am the example."), nl,  
3     writef("Talk to me."), nl, nl,  
4     repeat,  
5     writef("> "),  
6     getslist(X),  
7     respond(X),  
8     fail.
```

```
1 respond([bye]) :- known(name(X)), writef("Bye, "),
2               writef(X), nl, abort.
3 respond([bye]) :- writef("Bye!"), nl, abort.
4 respond([my,name,is,X]) :-
5               !, check(known(name(X))),
6               writef("Hello, "), writef(X), nl.
7
8 respond(_) :- writef("I didn't understand you."),
9               nl.
10
```

```
1 respond([what,is,my,name]) :-  
2     !, known(name(X)) ->  
3         (writef("Your name is "),  
4           writef(X), nl, !);  
5         !, writef("I don't know."), nl.
```

```
1 respond([X,is,Y]) :- !, L =.. [Y,X], check(L).
2 respond([forget,that,X,is,Y]) :-
3     !, L =.. [Y,X], forget(L).
4 respond([is,X,Y]) :-
5     !, L =.. [Y,X],
6     ((call(L), !, writef("Yes."), nl);
7     writef("I don't know."), nl).
8 respond([who,is,Y]) :-
9     L =.. [Y,X], !,
10    findall(X,L,Xs),
11    write(Xs).
```

```
1 respond([all,X,are,Y]) :- !,  
2     C1 =.. [Y,Z],  
3     C2 =.. [X,Z],  
4     L =.. [(:-),C1,C2],  
5     check(L),  
6     writef("Okay."), nl.  
7  
8 respond([what,do,you,know,about,X]) :- !,  
9     findall(Y, (aclause(Y),  
10             L =.. [Y,X], call(L)), Xs),  
11     write(Xs).
```